CODD’S RULES SQL QUERIES

Rule 1: The Information Rule

All data in the database is stored in tables consisting of columns and rows. Personal details for all patients are stored within discrete cells in the ‘patients’ table.

```
SELECT first_name FROM patients WHERE patient_id=101;
```

The first name of any patient can be retrieved from its unique location using the ‘patient_id’ (primary key) and the ‘first_name’ column.

Rule 2: The Guaranteed Access Rule

All data is accessible by combining table name, column name and unique primary key that identifies each row. Consequently, each row must have a primary key.

```
SELECT dentist_fname, dentist_surname FROM dentist WHERE dentist_id=102;
```

A dental specialist’s name can be accessed by referencing the column names (dentist_fname, dentist_surname), the row (dentist_id) and the table name (dentist).

Rule 3: Systematic Treatment of Null Values

Unknown, missing or not applicable data is consistently represented as a NULL value distinct from a zero value or empty string. Logical and arithmetical operations involving NULL will always return NULL. Primary keys are never NULL.

```
SELECT invoice.invoice_id, payment_method.amount_paid AS TotalPaid, payment_status
 FROM invoice LEFT JOIN payment_method ON payment_method.invoice_id = invoice.invo
ice_id
```

An invoice can have zero or many payments. If an invoice has zero payments, then the \`amount paid\` in the payment methods table does not exists for that invoice so is equal to NULL. This is distinct from a payment transaction with an \`amount paid\` totalling zero (i.e., in the case of a cancelled/declined payment transaction).
As other fields also do not have data, they also have NULL values even if they are not the same data type as \`amount paid\` field. The \`payment status\` field which is an ENUM (string object data type) is also NULL.

Rule 4: Dynamic Online Catalogue based on the relational model

The structure and format description of the entire database is stored in the relational format along with the database. It is accessible using the same query language used to access the database.

```
SELECT * FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_SCHEMA="dentistDB";
```

Any authorised user can access the metadata for the database using the above query. Information such as the name of tables and views, data types and creation dates are recorded in the same format as the data.

Rule 5: The Comprehensive Data Sub Language Rule

The database is accessed via a relational language (i.e., SQL) either directly or indirectly using its linear syntax. The data sublanguage must support data definition, manipulation, constraints and transaction management operations.

Data definition language example (Create a table for patient addresses, add constraints and comments)

```
-- Table Structure for patient addresses
CREATE TABLE addresses(
      address_id smallint(6) NOT NULL,
      street_name varchar(50) NOT NULL,
      town_name varchar(50) NOT NULL,
      county_name varchar(50) NOT NULL,
      eircode varchar(7) NOT NULL
      );

-- Add primary keys and constraints
ALTER TABLE addresses
ADD PRIMARY KEY (address_id),
MODIFY address_id smallint(6) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=101;
ADD CONSTRAINT addresses_ibuc_1 UNIQUE (eircode);
```

Data manipulation language example (Queries on the addresses table)

```
DELETE FROM addresses WHERE address_id=108;

INSERT INTO addresses(address_id, street_name, town_name, county_name, eircode) VA
LUES (108, '21 Main Street', 'Macroom', 'Cork', 'C54 EA31')

SELECT street_name, town_name, eircode FROM addresses WHERE county_name="Cork"

UPDATE addresses SET street_name="44 Bridge Street", town_name="Rockgrove"
WHERE address_id=108
```

Rule 6: The View Updating Rule

Views are virtual tables created from a stored query of the data that give a user a bespoke subset of its structure. All views that are theoretically updateable are also updateable by the system (Codd, 1985a).

For a view to be updateable it must meet several criteria including having a one-to-one relationship with the records in the underlying tables and without aggregate functions, left or outer joins, GROUP BY or HAVING clauses, etc.

The below view is created to present data from two tables `treatment_details` and `referral_details`. An update query is performed on this view and the changes are reflected in the view and also propagated to the base table.

```
CREATE VIEW `View Treatment By Speciality` AS(
SELECT treatment_name, treatment_description, speciality
FROM treatment_details INNER JOIN referral_details
ON referral_details.referral_id=treatment_details.referral_id);


UPDATE `View Treatment By Speciality`
SET treatment_name="Dental X-ray", treatment_description="Standard dental X-ray"
WHERE treatment_name="X-ray";
```

Rule 7: High Level Insert Update and Delete Rule

Data insertion, update and deletion operations that can be performed on a single row must also be possible on multiple rows through a single query.

Inserting multiple rows of new patient records into the ‘patients’ table.

```
INSERT INTO patients(address_id, first_name, last_name, birth_date, contact_number
, email, gender) VALUES
(108, 'George', Wilson', "1999-08-17", "0875468971", 'wilsons99@yahoo.com',
'male'),
(101, 'Sam', 'Smith', "2011-03-24", "0868642197", 'sammySthy_2011@gmail.com',
'female'),
(107, 'Maria', 'Alonso', "1995-03-14", "0868642197", 'maria.alonso@outlook.com',
'female')
```

Rule 8: Physical Data Independence

The physical storage location of the data should be independent of the logical level (i.e., tables, columns and rows) and how the data can be accessed logically.

Steps to move the database data from one local drive to another. This step also illustrates that the data can be transferred from one type of storage device to another (i.e., from an SSD to an HDD).

1. Check the database data location using:
   ```
   SELECT @@datadir
   ```
2. Stop Apache and MySQL modules in XAMPP and close XAMPP
3. Move data stored at: C:/xampp/mysql/data to new storage location: E:/Users/PJ/CoddsRule8
4. Change the file path saved within the configuration file found at C:/xampp/mysql/bin.my.ini from: “datadir=C:/xampp/mysql/data” to “datadir=E:/Users/PJ/CoddsRule8/data”
5. Open XAMPP and restart Apache and MySQL modules in XAMPP
6. Check the database data location again using:
   ```
   SELECT @@datadir
   ```

Rule 9: Logical Data Independence

Changes to the logical structure of tables should not impact the user’s view of the data. For example, data that can be split between two tables should return the same data (by using a join query between the two tables) as if it was still a single table. Also, addition/deletion of columns or tables that are not part of a user’s view should not affect that user’s view.

```
-- Create a single table for dentist details
Create TABLE dentistA (
      dentist_id smallint(6) NOT NULL,
      clinic_id smallint(6) NOT NULL,
      dentist_name char(15) NOT NULL,
      clinic_name varchar(50) NOT NULL,
      street_name varchar(50) NOT NULL,
      town_name varchar(50) NOT NULL,
      county_name varchar(50) NOT NULL,
      PRIMARY KEY (dentist_id)
      );

-- Create a view for User A showing all the data in table `dentistA`

CREATE VIEW `UserA` AS( SELECT * FROM dentistA)

-- Split the dentist data between two tables `dentistB` and `clinicB`:
CREATE TABLE clinicB (
      clinic_id smallint(6) NOT NULL,
      clinic_name varchar(50) NOT NULL,
      street_name varchar(50) NOT NULL,
      town_name varchar(50) NOT NULL,
      county_name varchar(50) NOT NULL,
      PRIMARY KEY (clinic_id)
      );

Create TABLE dentistB (
      dentist_id smallint(6) NOT NULL,
      clinic_id smallint(6) NOT NULL,
      dentist_name char(15) NOT NULL,
      PRIMARY KEY(dentist_id),
      FOREIGN KEY(clinic_id) REFERENCES clinic(clinic_id)
      );


-- Create an identical view as User A by using a join on the two separate tables:
CREATE VIEW `UserB` AS(
SELECT dentist_id, dentistB.clinic_id, dentist_name, clinic_name, street_name, tow
n_name, county_name FROM dentistB
INNER JOIN clinicB ON clinicB.clinic_id=dentistB.clinic_id);
```

Rule 10: Integrity Independence

Integrity constraints are defined using the relational data sublanguage and stored with the database in the data dictionary (i.e., database must be able to enforce and modify its own data integrity constraints independent of application programs).

- Entity integrity requires each table have its own non-NULL identifier primary key that is unique for each row.
- Referential integrity maintains a consistent relationship between tables. A foreign key must match the primary key that it references and all alterations to the primary key must also apply to all its foreign keys or else the transaction must be prevented.

Two tables are created using the SQL language, alongside them are the integrity constraints for primary keys and foreign keys. The presence of the foreign key in the `card_payment` table and the corresponding unique, non-NULL primary key in `payment_methods` meets the criteria for both referential integrity and entity integrity.

```
-- Create table for card payments
CREATE TABLE card_payment (
      card_id smallint(6) NOT NULL,
      payment_id int(11) NOT NULL,
      holder_name varchar(25) NOT NULL,
      card_number int(10) NOT NULL,
      exp_date date NOT NULL,
      card_type ENUM('VISA', 'Mastercard') NOT NULL
      );

-- Create table for card methods
CREATE TABLE payment_method (
      payment_id int(11) NOT NULL,
      invoice_id int(11) NOT NULL,
      payment_date datetime,
      payment_type ENUM('card', 'cheque', 'cash') NOT NULL,
      amount_paid decimal(10,2),
      payment_status ENUM('approved', 'declined', 'cancelled') DEFAULT NULL
      );

-- Create constraints for card payments table
ALTER TABLE card_payment
ADD PRIMARY KEY (card_id),
MODIFY card_id smallint(6) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=101,
ADD CONSTRAINT card_payment_ibfk_1 FOREIGN KEY (payment_id) REFERENCES payment_met
hod(payment_id);

-- Create constraints for card methods table
ALTER TABLE payment_method
ADD PRIMARY KEY (payment_id),
MODIFY payment_id int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=101,
ADD CONSTRAINT payment_method_ibfk_1 FOREIGN KEY (invoice_id) REFERENCES invoice(i
nvoice_id);
```

Rule 11: Distributed Independence

The database should always appear centralised to the user even if the data is part of a distributed system over a computer network or on multiple physical systems.

Being able to partition database components without impact on users is important in database scalability and performance as organisations expand globally. As well as adhering to data protection laws that require certain data is stored in its country of origin.

Dr Mulcahy will open a second dental practice in a neighbouring town. She will use a distributed data system. Patients attending a clinic will have their personal details, treatment records and payment information stored at that dental practice but Helen/Dr Mulcahy will be able to access all data from either clinic as if they were accessing it locally. This type of data fragmentation will reduce data redundancy.

Rule 12 Non-Subversion Rule

If a low-level language interface is allowed, it must not circumvent the structural integrity of the database (e.g., bypass security or integrity constraints of the relational language). Only the relational language that defined the integrity constraints should be able to redefine them.

As more and more databases are distributed over a network, cybersecurity is becoming more important. Any DBMS that does not adhere to rule 12 leaves a potential gap in security.

Many DBMS violate this rule by allowing bulk loaders to insert data while disabling integrity constraints and triggers. This compromises the integrity of the data.

When merging with another dental clinic the DBMS only permits patient records for the partner clinic added to the database using SQL.

References:

Codd, E. F., “Is Your DBMS Really Relational?”, ComputerWorld (1985a).

Codd, E. F., “Does Your DBMS Run By the Rules”, ComputerWorld (1985b).

Connolly, T. and Begg, C., 2014. Database Systems A Practical Approach to Design, Implementation and Management, 6th Ed., Pearson Education

Date, C.J., 2003. An Introduction to Database Systems, 8th Ed., Person Education.

Davidson, L., Kline, K., Klein, S. and Windisch, K., 2008. Pro SQL Server 2008 Relational Database Design and Implementation, Apress, Appendix A

Jethwa, J., 2017. Change MySQL Data directory location in Windows [Online]. Available from: https://dbatricksworld.com/change-mysql-data-directory-location-in-windows/

Rel 2019 Codd’s Twelve Rules [Online]. Available from: https://reldb.org/c/index.php/twelve-rules/

MySQL 8.0 Reference Manual, 2021 [Online]. Available from: https://dev.mysql.com/doc/refman/8.0/en/